Data Structure : ALGORITHMS

What?

Algorithm

A set of Step by Step Instructions that provide a solution to a problem

"a finite sequence of well-defined, computer-implementable instructions"

- Wiki

"An algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output"

-Coreman

Characteristics

- Step By Step
- Un-Ambiguous
- Finite
- Feasible
- Independent

Data Structure : ALGORITHMS

It is important to –
plan a pathway to solve problems before implementing solutions
And
To solve such problem in modular way

modular way means to break a complex problem in smaller problem (modules)
Advantage:

Easy to Plan
Easy to implement
Easy to understand
Easy to debug (find errors)
Easy to Test

This can be done using ALGORITHMS

Data Structure : ALGORITHMS

During Planning

INPUT → **Algorithm** → OUTPUT

Written in Natural Language

Question : This is a program/process also do. Then What is the Difference?

During Compilation

INPUT → **Program** → OUTPUT

Written in Programming Language

During Runtime

INPUT → **Process** → OUTPUT

Written in Binary Language

Data Structure : ALGORITHMS

Algorithm

can be written in

in a general language that is easily understandable

Natural Language like English

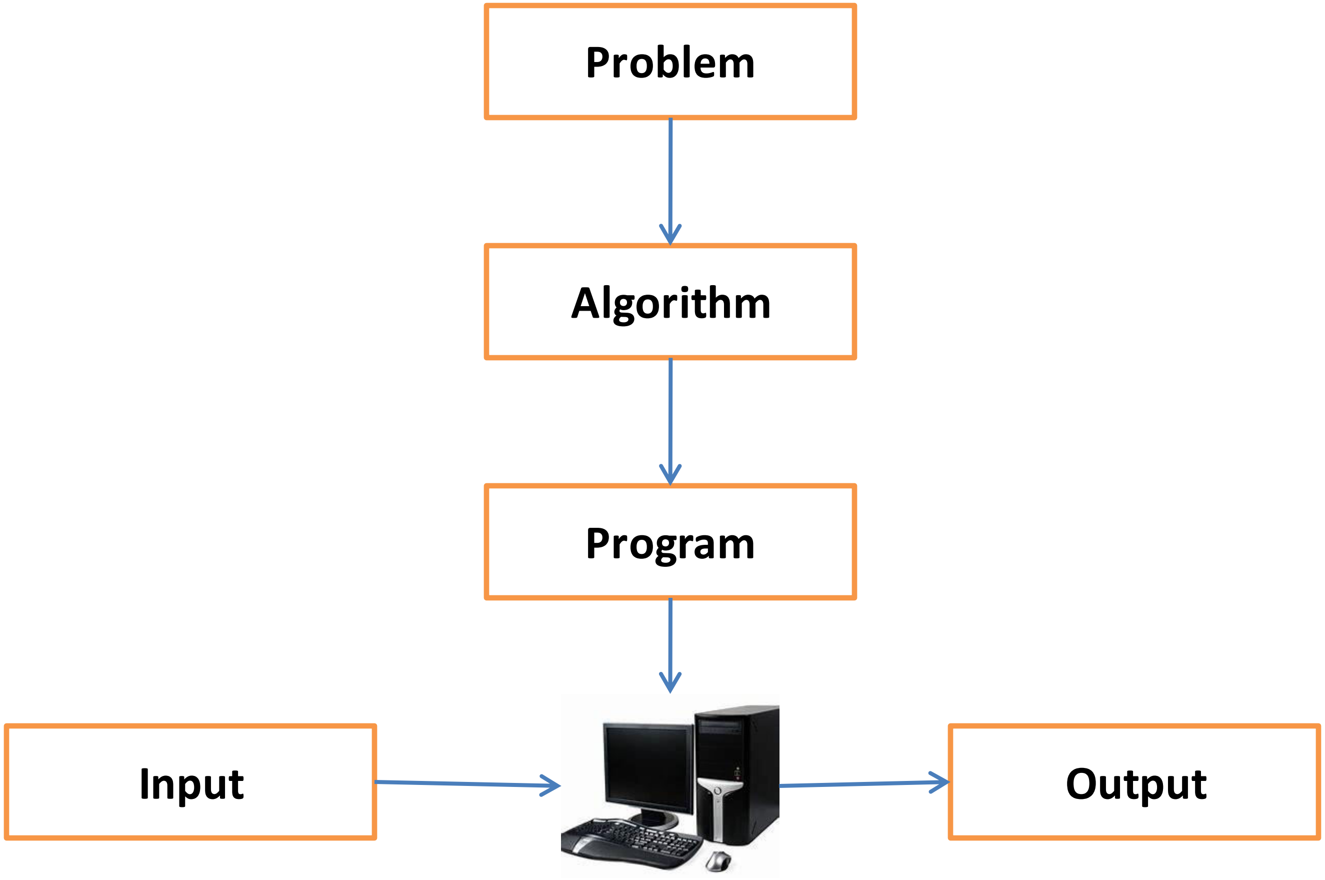
It is a convention to use Pseudo code or Flow Charts etc.

Symbolic Instruction

Diagrammatic Representation

Data Structure : ALGORITHMS

An ***algorithm is a sequence of unambiguous instructions for solving a*** problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.



- The non ambiguity requirement for each step of an algorithm cannot be compromised.
- The range of inputs for which an algorithm works has to be specified carefully.
- The same algorithm can be represented in several different ways.
- There may exist several algorithms for solving the same problem.

Algorithms for the same problem can be based on very different ideas and can solve the problem with dramatically different speeds.

Data Structure : ALGORITHMS

One Problem Many Solution

Algorithm Name: Buy_new_TV (Brand, Model)
Input: Company name as TV Brand, Model No as Model
Output: You are a proud owner of a TV

Start
1. Visit Website
2. Select Brand
3. Select Model
4. Internet Banking

End

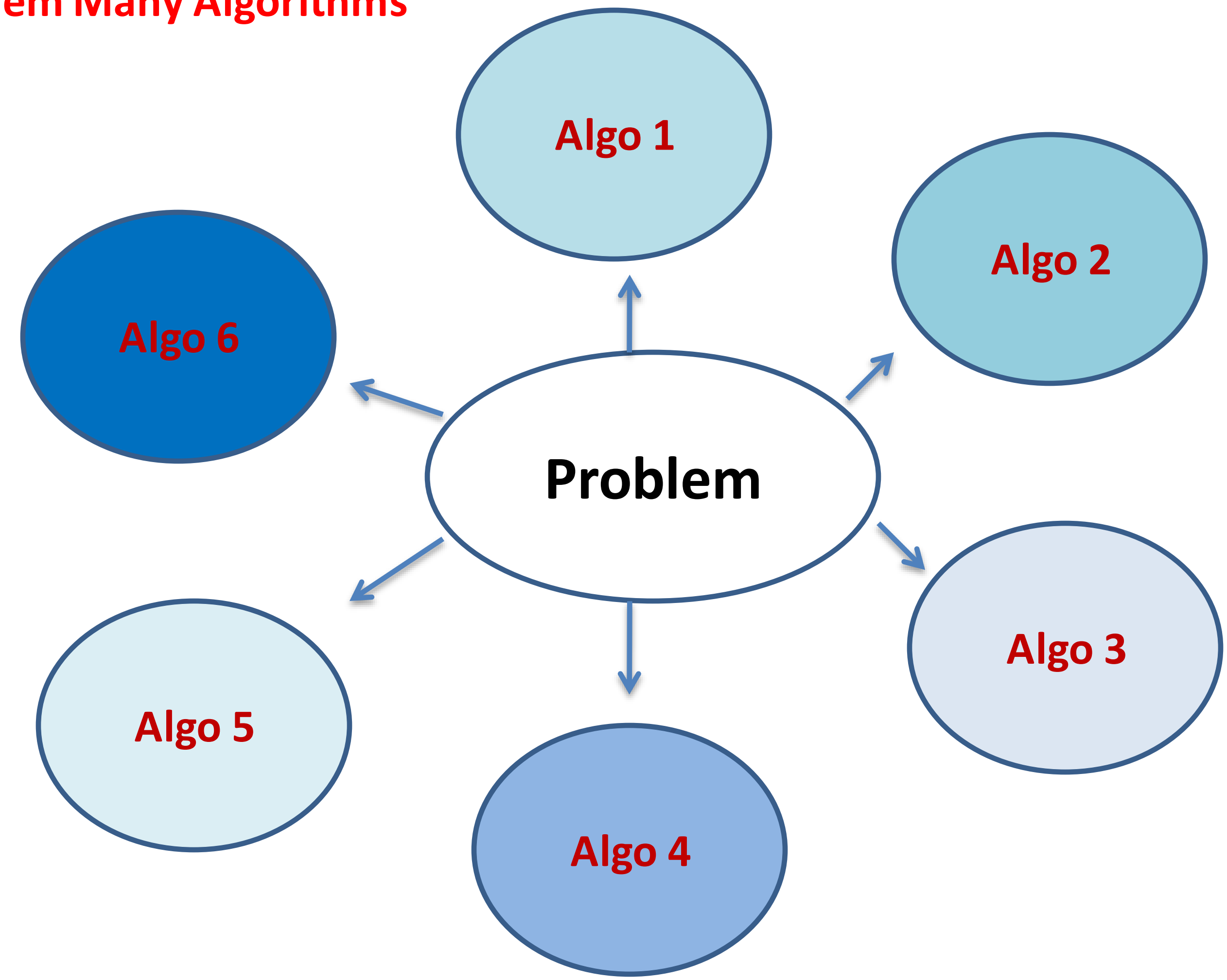
Start
1. Visit Website
2. Select Brand
3. Select Model
4. Credit Card

End

Start
1. Visit Website
2. Select Brand
3. Select Model
4. Debit Card

End

One Problem Many Algorithms



Data Structure : ALGORITHMS

How to Choose Best suitable Algorithm for our problem?

Algorithm Analysis

Priori Analysis	Posterior Analysis (Empirical, Experimental)
Before actually using algorithm	After actually using algorithm
Theoretical	Practical
Implementation-> not Required	Implementation -> Required

Priori Analysis

Advantage:
- Fast
- No extra Resources required

Disadvantage:
- Based on assumptions

Posterior Analysis

Advantage:
- Accurate and Actual

Disadvantage:
- Implementation of all algorithms is an overhead

Algorithm Analysis

Space Complexity Analysis

How much Space Required?

Time Complexity Analysis

How much Time Required?

Space Complexity (Algorithm A) =

Space Complexity (Compile Time) + Space Complexity (Run Time)

SC(A) = c + SC (I)

Constant (c)

- Variables
- Objects
- Dynamic Memory allocated

Time Complexity (Algorithm A) =

Time taken by one operation * No. of operations

Data Structure : ALGORITHMS

Priori Analysis / Theoretical Analysis

ALGO 1

ALGO 2

Problem

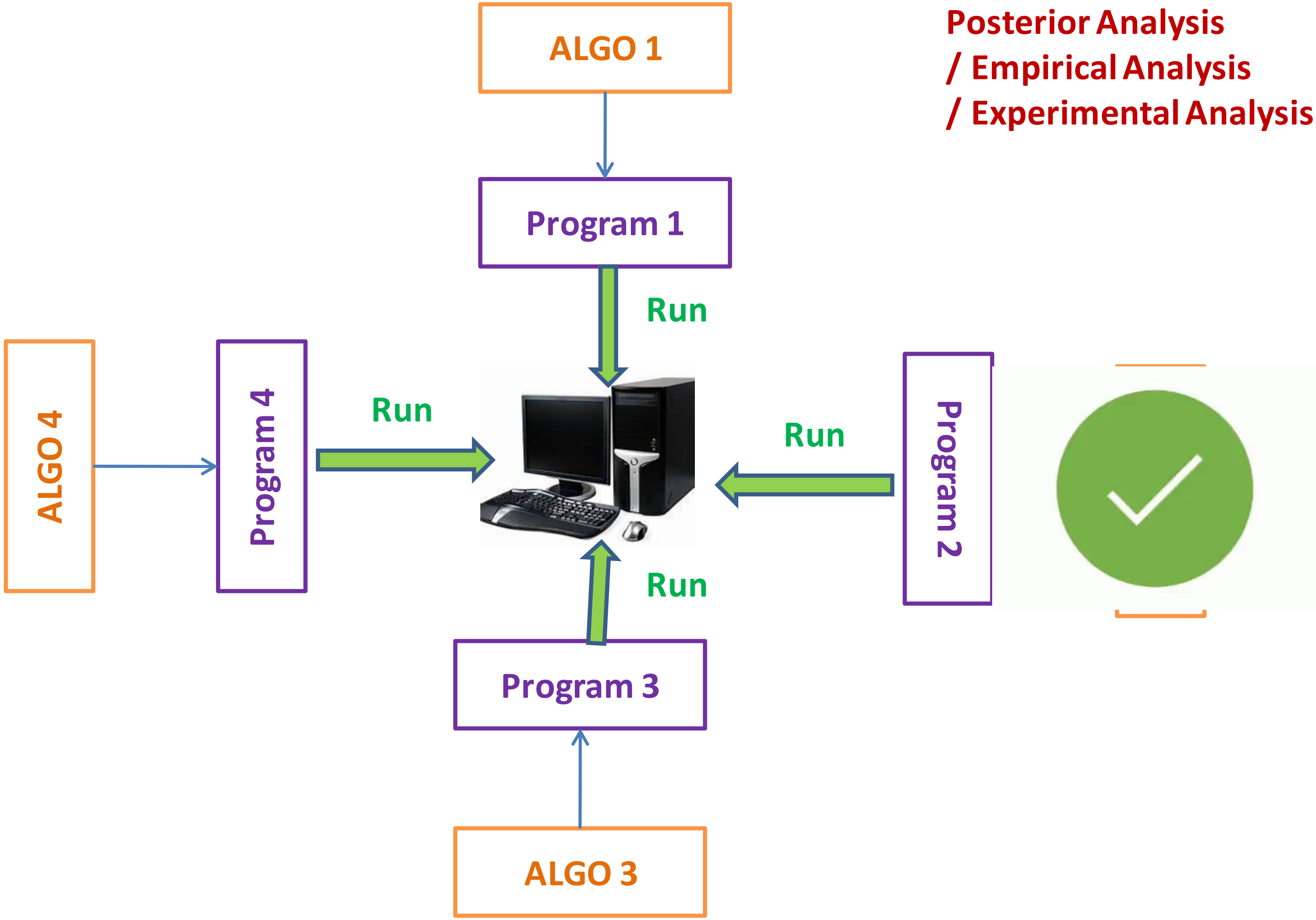
ALGO 3

ALGO 4

…………………	…………………
…………………	…………………
…………………	…………………
Algo1	Algo2
SC …..	SC …..
TC …..	TC …..
…………………	…………………
…………………	…………………
…………………	…………………
Algo3	Algo4
SC …..	SC …..
TC	TC …..

Data Structure : ALGORITHMS

Posterior Analysis
/ Empirical Analysis
/ Experimental Analysis



Data Structure : ALGORITHMS

Platform Dependency

PROBLEMS

Posterior Analysis
/ Empirical Analysis
/ Experimental Analysis

ALGO 2

Program 1

Run

32 BITS

64 BITS

SC .....
TC .....

SC .....
TC .....


AMD
ATHLON
PRO


intel
Core 2 Quad

OS Dependency

PROBLEMS

ALGO 2

Program 1

Run

SC .....
TC .....

SC .....
TC .....

SC .....
TC .....

SC .....
TC .....

SC .....
TC .....

Posterior Analysis
/ Empirical Analysis
/ Experimental Analysis

Data Structure : ALGORITHMS

Language Dependency

PROBLEMS

ALGO 2

Program 1

Run

SC .....
TC .....

SC .....
TC .....

SC .....
TC .....

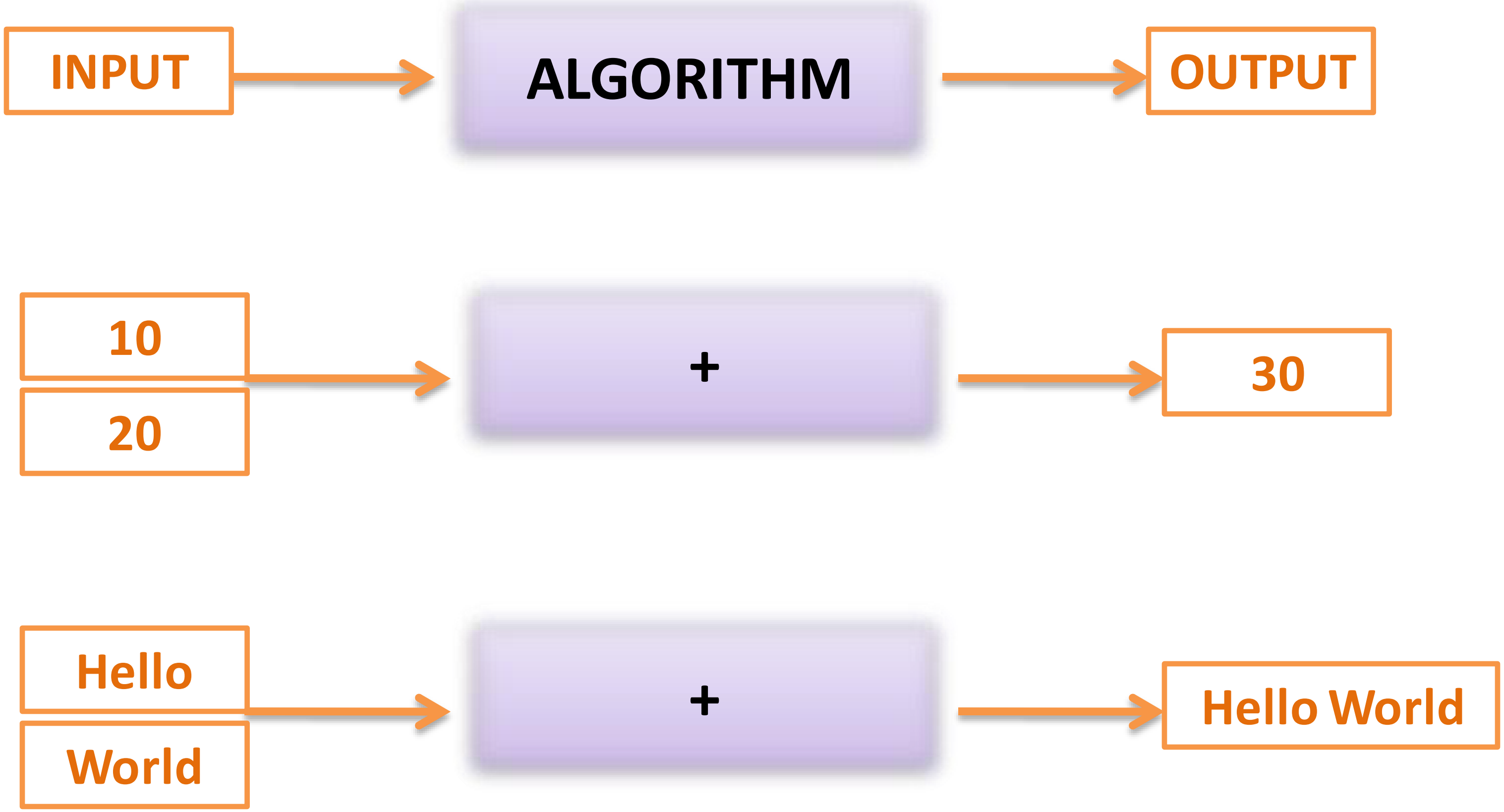
SC .....
TC .....

SC .....
TC .....

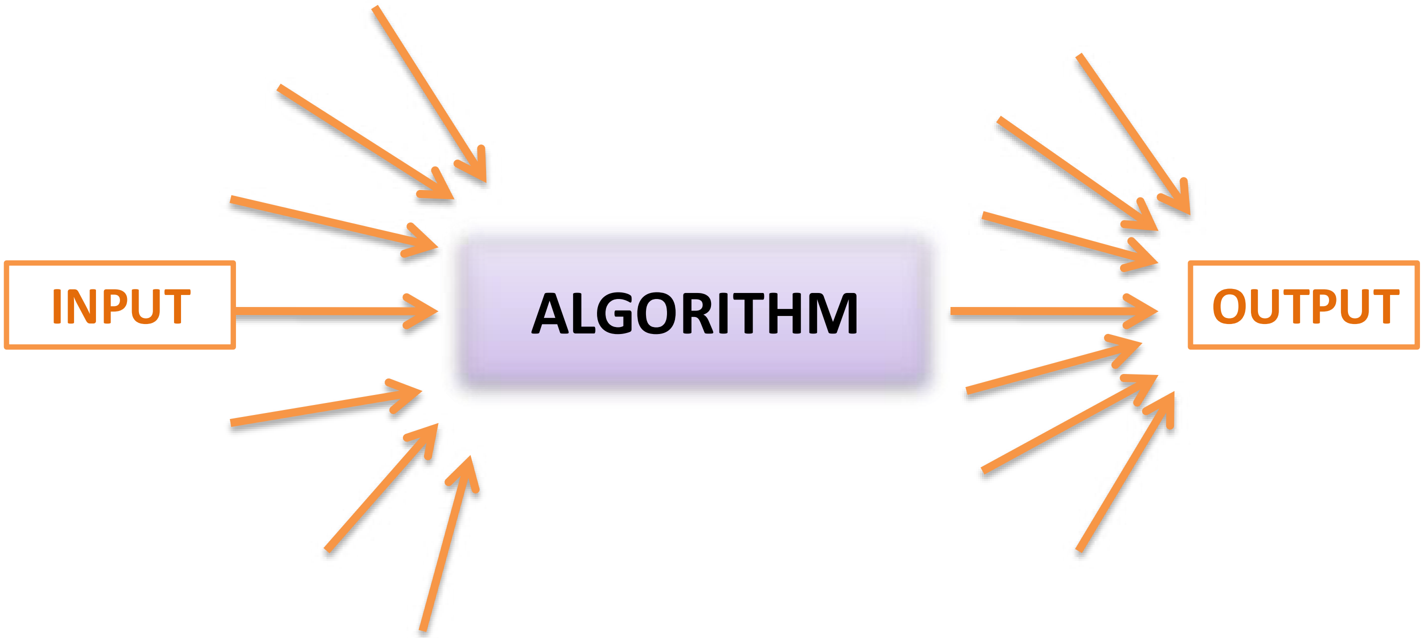
Posterior Analysis
/ Empirical Analysis
/ Experimental Analysis


Self
Ioke
Ada
alice
Io
prolog
F#
php
pure
BOO
Ftn
Clean
JavaScript
subtext
Dylan
ELAN
Gosu
Sed
BASH
Rexx
Asymptote
C#
SQL
ALLOY
Turing
Orc
Jif

One Algorithm Many Instances

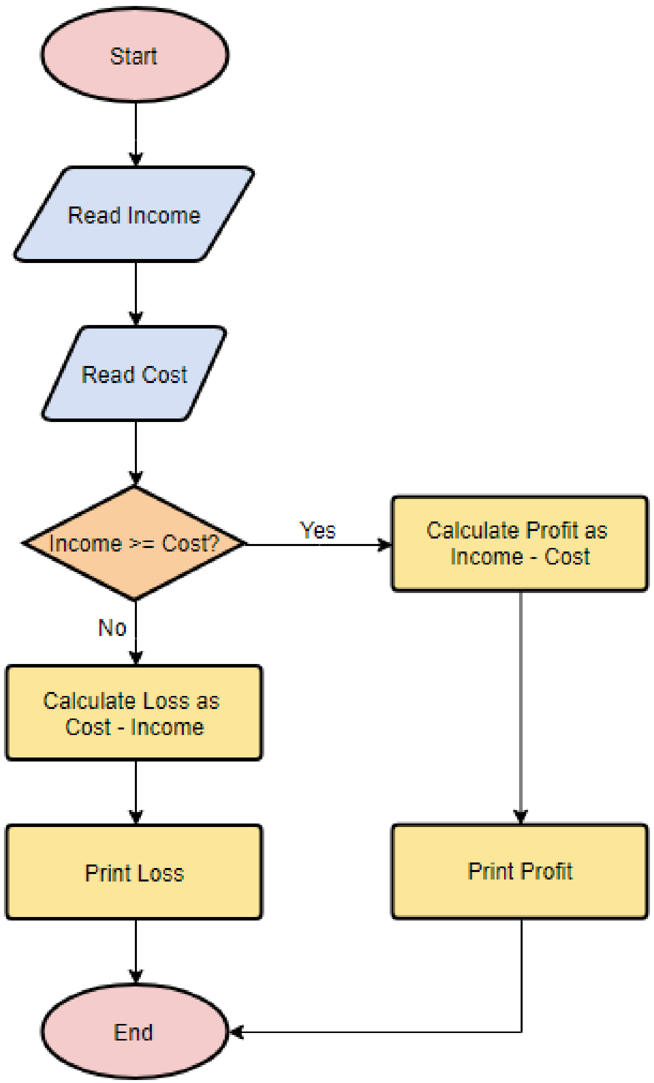


An input to an algorithm specifies an instance of the problem the algorithm solves

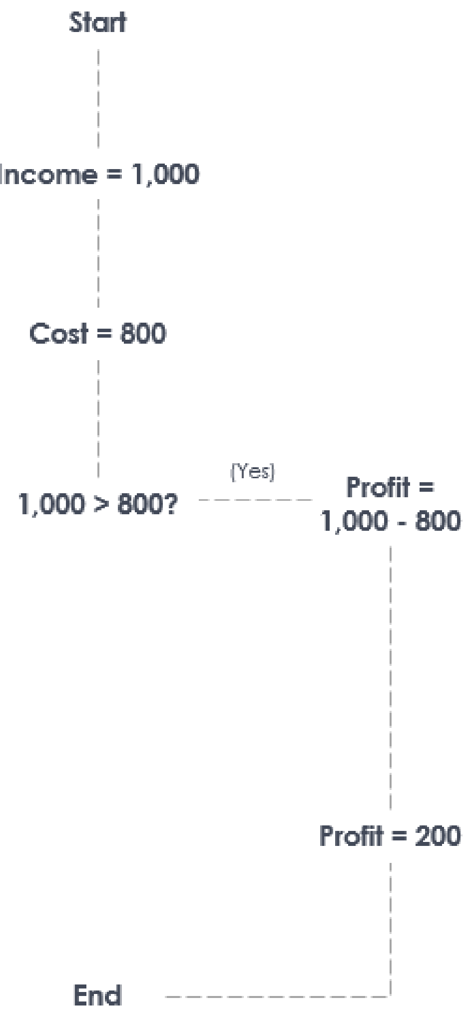


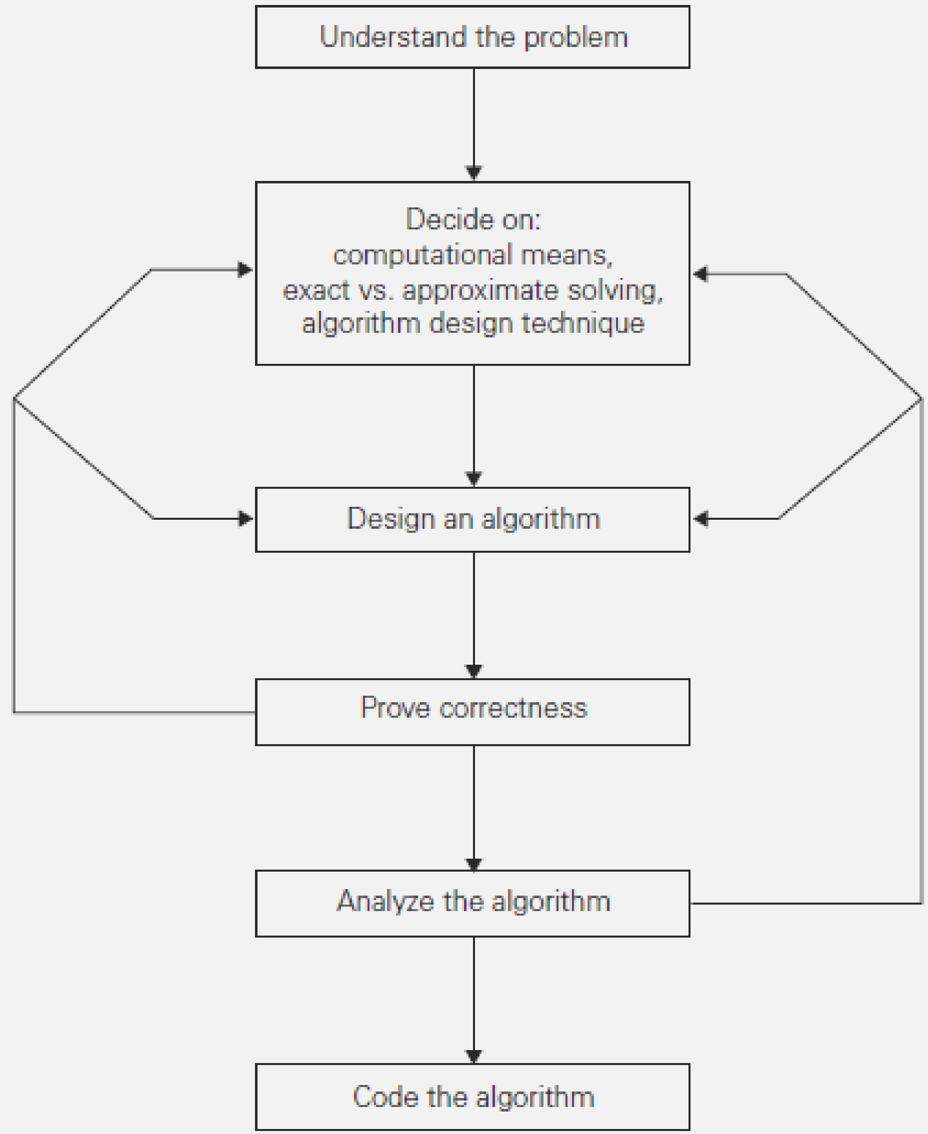
It is very important to specify exactly the set of instances the algorithm needs to handle.

Data Structure : ALGORITHMS



Find the profit/loss when income = 1,000, cost = 800





IGURE 1.2 Algorithm design and analysis process.

Understanding the Problem Completely

First Step before designing an algorithm

- Read the problem's description carefully
- Ask questions if you have any doubts about the problem
- Do a few small examples by hand
- think about special cases

If you fail to do this, your algorithm may work correctly for a majority of inputs but crash on some "boundary" values.

Remember that-

"A correct algorithm is not one that works most of the time, but one that works correctly for *all legitimate* inputs"

Ascertaining the Capabilities of the Computational Device

The vast majority of algorithms in use today are still destined to be programmed for a computer closely resembling –

"von Neumann machine"—a computer architecture outlined by the prominent Hungarian-American mathematician John von Neumann (1903–1957), in collaboration with A. Burks and H. Goldstine, in 1946.

Based on Random-access machine (RAM)

Its central assumption is that instructions are executed one after another, one operation at a time. Accordingly, algorithms designed to be executed on such machines are called ***sequential algorithms.***

1. ***************
2. ***************
3. ***************
4. ***************

The central assumption of the RAM model does not hold for some newer computers that can execute operations concurrently, i.e., in parallel. Algorithms that take advantage of this capability are called ***parallel algorithms.***

1. ***************
2. ***************
3. ***************
4. ***************

Choosing between Exact and Approximate Problem Solving

The next principal decision is to choose between –

Solving the problem exactly - an ***Exact Algorithm***
Solving the problem approximately - ***Approximation Algorithm***

Why Approximate Algorithms?

First, there are important problems that simply cannot be solved exactly for most of their instances

Extracting Square Roots
Solving Nonlinear Equations
Evaluating Definite Integrals

Second, available algorithms for solving a problem exactly can be unacceptably slow because of the problem's intrinsic complexity.

Algorithm Design Techniques

An ***algorithm design technique (or "strategy" or "paradigm") is a general*** approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

Branch and Bound Algorithms
Greedy Algorithms
Divide and Concur

Learning these techniques is of utmost importance for the following reason-

They provide guidance for designing algorithms for new problems, i.e., problems for which there is no known satisfactory algorithm.

Designing an Algorithm and Data Structures

One should pay close attention to choosing data structures appropriate for the operations performed by the algorithm

Algorithms + Data Structures = Programs

Wirth, N. *Algorithms + Data Structures = Programs. Prentice-Hall,* Englewood Cliffs, NJ, 1976.

Methods of Specifying an Algorithm

flowchart

Pseudo code

Proving an Algorithm's Correctness

CORRECTNESS

The algorithm yields a required result for every legitimate input in a finite amount of time

Mathematical Induction

"It might be worth mentioning that although tracing the algorithm's performance for a few specific inputs can be a very worthwhile activity, it cannot prove the algorithm's correctness conclusively

But in order to show that an algorithm is incorrect, you need just one instance of its input for which the algorithm fails"

Analyzing an Algorithm

Efficiency

There are two kinds of algorithm efficiency:

Time Efficiency: *indicating how fast the algorithm runs*

Space efficiency: indicating how much extra memory it uses

Simplicity ***Generality***

Coding an Algorithm

An inefficient implementation can diminish your algorithm's power

Compilers do provide "Code Optimization"

Standard Tricks for Code Tuning –

- Computing a loop's invariant outside the loop
- Collecting common sub expressions
- Replacing expensive operations by cheap ones, and so on

Kernighan, B.W. and Pike. R. *The Practice of Programming.* Addison-Wesley, 1999.

Bentley, J. *Programming Pearls, 2nd ed. Addison-Wesley, 2000.*

Fundamentals of the Analysis of Algorithm Efficiency

Not everything that can be counted counts, and not everything that counts can be counted.

—Albert Einstein (1879–1955)